WHAT IS CLAIMED IS:

1. A method of profiling code for an execution environment in which latency exists between an execution event and detection thereof, the method comprising:

executing the code;

detecting the execution event; and

backtracking from a point in the code coinciding with the detection to a preceding operation associated with the execution event, the backtracking identifying the preceding operation at an expected displacement from the detection point unless an ambiguity creating location is disposed therebetween.

2. The method of claim 1,

wherein the ambiguity creating location is a branch target location.

3. The method of claim 2,

wherein ambiguity otherwise associated with at least some branch target locations is bridged using branch history information.

4. The method of claim 1,

wherein the ambiguity creating location is an entry point location.

5. The method of claim 1, wherein the ambiguity creating location is one of:

a jump target location;

an indirect branch target location;

a trap handler location; and

an interrupt handler location.

6. The method of claim 1,

wherein the preceding operation corresponds to a load instruction; and

wherein the execution event is a cache miss.

7. The method of claim 1,

wherein the preceding operation corresponds to a memory access instruction; and

wherein the execution event is either a hit or a miss at a level in a memory hierarchy.

8. The method of claim 1,

wherein the execution event is either a overflow or an underflow of a hardware counter.

9. The method of claim 1,

wherein the execution event triggers either a overflow or an underflow that is itself detected.

10. The method of claim 1,

wherein the latency includes that associated with delivery of a trap.

11. The method of claim 1,

wherein the latency includes that associated with delivery of a counter overflow event signal.

12. The method of claim 1,

wherein the latency is associated with pipeline execution skid.

13. The method of claim 1,

wherein the latency is associated with completion of in-flight operations.

14. The method of claim 1, embodied in a computer program product.

15. The method of claim 1, embodied in at least one of:

a profiling tool;

a code optimizer; and

a runtime library.

16. The method of claim 1, employed in combination with a compiler that pads the code with one or more padding operations to absorb at least some instances of the latency.

17. The method of claim 16,
wherein the padding operations are not themselves associated with the execution event.

18. The method of claim 16,
wherein the padding operations are not themselves ambiguity creating locations.

19. A method of identifying operations associated with execution events, the method comprising:
from a point in an execution sequence of the operations, the point coinciding with an execution event, backtracking through the operations toward a particular operation that precedes the coinciding point by an expected latency; and
associating the execution event with the particular operation unless the backtracking encounters an unresolved intervening target of a control transfer.

20. The method of claim 19, further comprising:
executing the sequence of operations on a processor; and
detecting the execution event.

21. The method of claim 19,
wherein the operations are instructions executable on a processor; and
wherein the particular operation is a particular one of the instructions that triggers the execution event.

22. The method of claim 19,
wherein the operations are executable on a processor and correspond to instructions of program code; and

wherein the particular operation corresponds to a particular one of the operations that triggers the execution event.

23. The method of claim 19,
wherein the execution event is an exception triggering execution of the particular operation.

24. The method of claim 19,
wherein the execution event is a cache miss.

25. The method of claim 19,
wherein the expected latency includes an trap delivery delay.

26. The method of claim 19,
wherein the execution event triggers a hardware event and the expected latency includes delivery of a signal associated therewith.

27. The method of claim 26,
wherein the hardware event is either underflow or overflow of a counter associated with the execution event.

28. The method of claim 19,
wherein the execution event is either underflow or overflow of a counter.

29. The method of claim 19,
wherein instances of intervening control transfer targets are identified in the execution sequence of operations to facilitate the backtracking.

30. The method of claim 29,
wherein at least some of the instances of intervening control transfer targets are resolved using branch history information.

31. The method of claim 19,

wherein control transfer target locations in the execution sequence are identified by a compiler.

32. The method of claim 19,
wherein the operations are instructions executable on a processor; and
wherein the particular operation is a particular one of the instructions that triggers the execution event.

33. The method of claim 19,
wherein the operations are executable on a processor and correspond to instructions of program code; and
wherein the particular operation corresponds to a particular one of the operations that triggers the execution event.

34. The method of claim 19, further comprising:
preparing the execution sequence of the operations.

35. The method of claim 34,
wherein preparation of the execution sequence includes identifying a location of the control transfer target therein.

36. The method of claim 34,
wherein preparation of the execution sequence includes identifying a location of the particular operation therein.

37. The method of claim 19,
wherein the particular operations include memory referencing instructions.

38. The method of claim 37,
wherein the memory referencing instructions include one or more of loads, stores and prefetches.

39. A method of associating an execution characteristic of code with a particular operation thereof, the method comprising:

identifying at least first-type and second-type operations in the code;

from a point in an execution sequence of the code that coincides with delayed detection of the execution characteristic, backtracking toward a candidate triggering operation of the first-type and associating the candidate triggering operation with the execution characteristic unless an unresolved intervening operation of the second-type is encountered.

40. The method of claim 39,

wherein the first-type operations include memory access operations.

41. The method of claim 39,

wherein the second-type operations include operations that coincide with control transfer target locations in the code.

42. The method of claim 39,

wherein the execution characteristic involves memory access latency.

43. The method of claim 39,

wherein the execution characteristic includes a cache miss statistic.

44. The method of claim 39,

wherein the detection delay includes a pipelined execution skid latency.

45. The method of claim 39,

wherein the second-type operations include operations that coincide with branch target locations in the code; and

wherein at least some of the branch target locations are resolved using branch history information.

46. A method of preparing code, the method comprising:

preparing a first executable instance of the code, the preparing identifying at least ambiguity creating locations therein;

executing the first executable instance and responsive to detection of an execution characteristic, backtracking through the code to identify an

associated operation thereof, wherein extent of the backtracking is limited at least by encountering of an unresolved intervening one of the identified ambiguity creating locations; and

further preparing a second executable instance of the code using the association between the associated operation and the execution characteristic.

47. The method of claim 46,

wherein the association between the associated operation and the execution characteristic is based on a statistically-significant set of additional detections and responsive backtracking.

48. The method of claim 46,

wherein the execution characteristic involves memory access latency; and

wherein the preparation of the second executable instance includes insertion of prefetch operations into the code.

49. The method of claim 46, further comprising:

resolving at least some intervening ones of the identified ambiguity creating locations using branch history information.

50. A computer program product encoded in one or more computer readable media, the computer program product comprising:

an execution sequence of operations; and

padding operations following at least some particular operations of the execution sequence, the padding operations providing an unambiguous skid region of the execution sequence.

51. The computer program product of claim 50,

wherein the particular operations include memory access operations.

52. The computer program product of claim 50,

wherein the padding operations include nops.

53. The computer program product of claim 50,
wherein the unambiguous skid region does not include an ambiguity creating location.

54. The computer program product of claim 50,
wherein the one or more computer readable media are selected from the set of a disk, tape or other magnetic, optical, semiconductor or electronic storage medium and a network, wireline, wireless or other communications medium.

55. A computer program product encoded in one or more computer readable media, the computer program product comprising:
an execution sequence of operations; and
one or more data sections that identify in the execution sequence at least ambiguity creating locations and target operations for use by one or both of a profiler and a optimizer.

56. The computer program product of claim 55,
wherein the ambiguity creating locations include branch target locations.

57. The computer program product of claim 55,
wherein the target operations include memory referencing operations.

58. The computer program product of claim 55,
wherein the one or more computer readable media are selected from the set of a disk, tape or other magnetic, optical, semiconductor or electronic storage medium and a network, wireline, wireless or other communications medium.

59. An apparatus comprising:
means for backtracking, from a point coinciding with an execution event in an execution sequence of operations, through the execution sequence toward a particular operation thereof that precedes the coinciding point; and

means for associating the execution event with the particular operation unless the backtracking encounters an intervening ambiguity creating location.

60. The apparatus of claim 59, further comprising:
means for bridging at least some ambiguity creating locations.

61. An apparatus comprising:
a code preparation facility suitable for preparation of an execution sequence of operations; and
means for padding the execution sequence to provide an unambiguous skid region therein.